
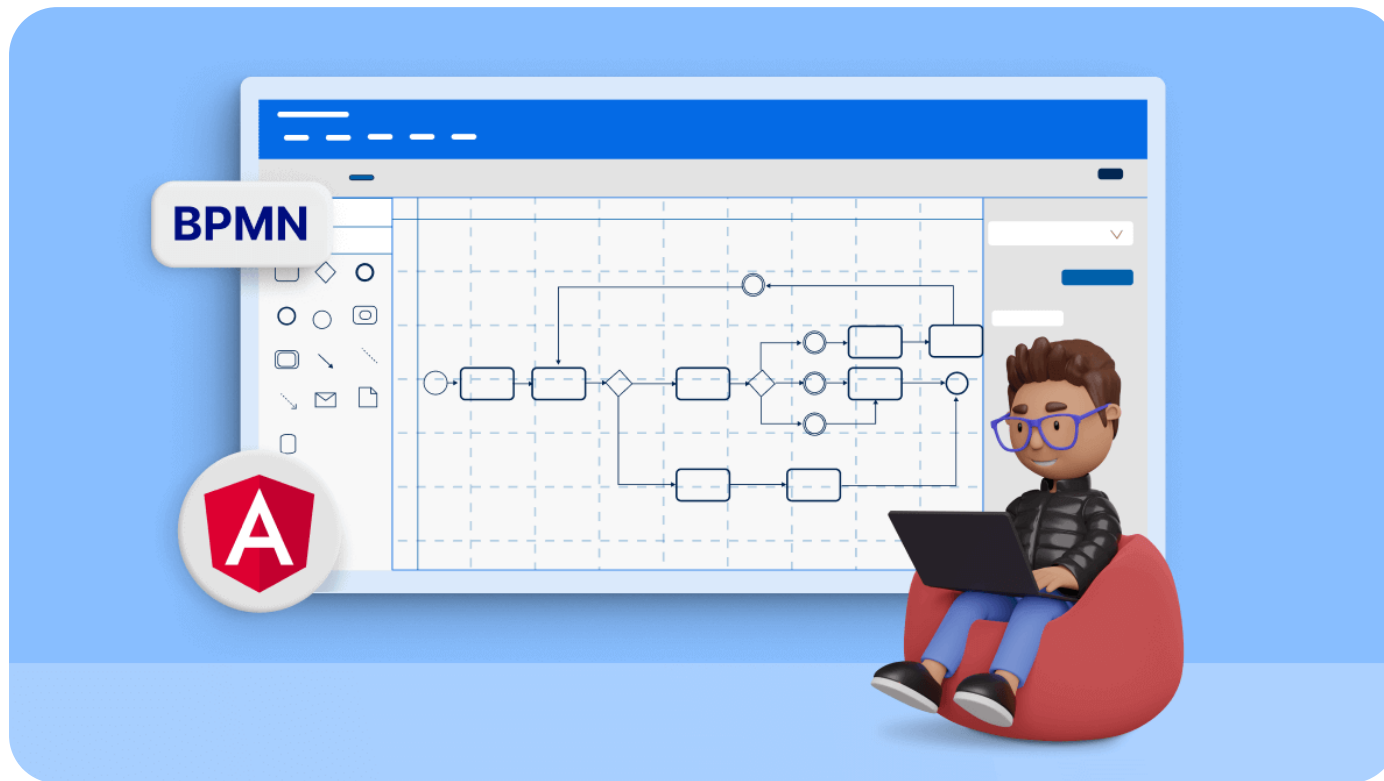


Easily Build an Interactive BPMN Viewer and Editor in Angular

 **Moulidharan Gopalakrishnan** •  16 min read •  Nov 18, 2024 • **Updated**



Build stunning
real world apps using

Angular Components

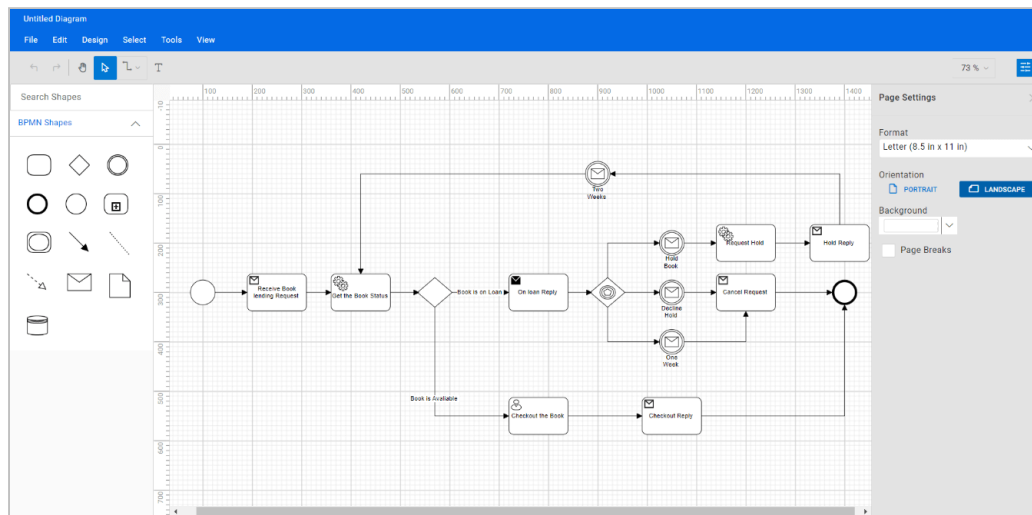
 Syncfusion®

Try it free

A [business process model and notation \(BPMN\)](#) diagram is used to visually represent a complex business process in a form like a flowchart, making it easier to understand for anyone not directly involved with the process.

This blog will show you how to create a BPMN viewer and editor with the interactive features of the Syncfusion [Angular Diagram control](#).

Let's get started!



Creating a BPMN Viewer and Editor Using the Angular Diagram Control

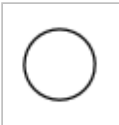
Prerequisites

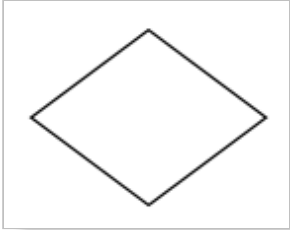
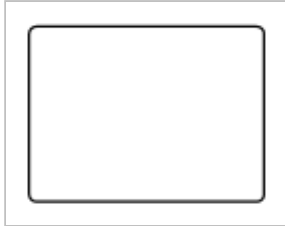
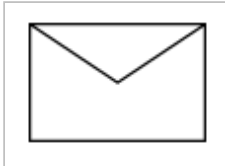
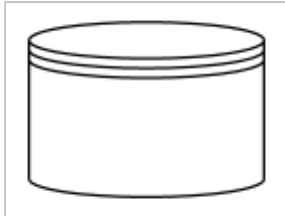
- Install [Node.JS version 14.17.3](#).

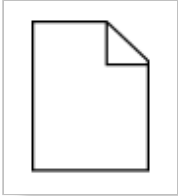
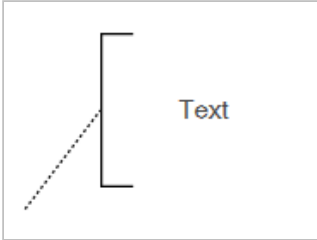

BPMN shapes





The BPMN shapes are used to represent internal business processes in a graphical notation. They provide a standard method for communicating procedures, making them a popular and intuitive graphic easily understood by all stakeholders, including business users, analysts, software developers, and data architects.



The Angular Diagram control supports the following BPMN shapes:


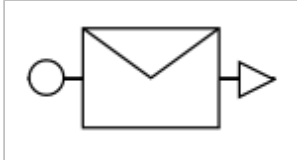
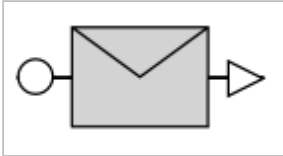
Shape	Symbol	Description
Event		Represents something that happens during a business process.

Gateway		Used to control the flow of a process.
Activity		Describes the kind of work being done in a particular process instance.
Message		The content of a communication.
DataStore		Store or access data associated with a business process.

DataObject		<p>Represents information flowing with the process.</p> <p>Examples: Data placed into a process, data resulting from a process, or data that needs to be collected or stored.</p>
TextAnnotation		<p>Points at or references another BPMN shape.</p>
ExpandedSubProcess		<p>The extended version of the Group symbol.</p>

SequenceFlow		Represents the typical path between two flow objects.
ConditionalSequenceflow		Controls the flow of a process based on certain conditions.
DefaultSequenceFlow		Represents the default sequence flow and that other conditional flows are not valid. The latter is indicated by the backslash at the beginning of the sequence.
Association		Represents a relationship between artifacts and flow objects.

DirectionalAssociation		A directional association is used with data objects to show that a data object is either an input or output from an activity. This is represented as a dotted graphical line with an arrow at one end.
BiDirectionalAssociation		A bidirectional association is used with data objects to show that a data object is both input and output from an activity. This is represented as a dotted graphical line with arrows at both ends.

MessageFlow		Shows the flow of messages between two participants.
InitiatingMessageflow		Represents an activity or event in one pool that can initiate a message to another pool.
NonInitiatingMessageflow		Represents an activity or event in one pool that can't initiate a message to another pool.

Creating the BPMN editor surface

Note: If you are new to our Angular Diagram control, please refer to the [Getting Started documentation](#) before proceeding.

Follow these steps to add the Syncfusion Angular Diagram control to your web app:

Step #1: Create a folder and name it **BPMN Editor**.

Step #2: Open your terminal or command prompt. Then, run the following command to install the Angular CLI version 14.2.9 on your system.

```
npm i @angular/cli@14.2.9
```



Step #3: Create a new Angular app named **BpmnEditorApp** using the following command. This command should be executed in your terminal or command prompt.

```
ng new BpmnEditorApp
```



Step #4: Next, navigate to the newly created project folder by executing the following command in your terminal or command prompt. This will change your current directory to the **BpmnEditorApp** directory.

```
cd BpmnEditorApp
```



Step #5: Install the Syncfusion Angular Diagram package using either of the following methods:

- The first method is to install the required NPM package using the following command in your terminal or command prompt.

```
npm install @syncfusion/ej2-angular-diagrams --save
```



- The second method is to add the Angular Diagram control and its dependency packages to your **package.json** file. Then, you can run the **npm install** command to install them.

```
"dependencies": {  
  "@angular/animations": "^14.2.0",  
  "@angular/common": "^14.2.0",  
  "@angular/compiler": "^14.2.0",  
  "@angular/core": "^14.2.0",  
  "@angular/forms": "^14.2.0",  
  "@angular/platform-browser": "^14.2.0",  
  "@angular/platform-browser-dynamic": "^14.2.0",  
  "@angular/router": "^14.2.0",  
}
```



```

"@syncfusion/ej2-angular-buttons": "^20.3.57",
"@syncfusion/ej2-angular-diagrams": "^20.4.40",
"@syncfusion/ej2-angular-dropdowns": "^20.3.57",
"@syncfusion/ej2-angular-inputs": "^20.4.42",
"@syncfusion/ej2-angular-navigations": "^20.4.43",
"@syncfusion/ej2-angular-popups": "^20.3.57",
"@syncfusion/ej2-angular-splitbuttons": "^20.3.56",
"rxjs": "~7.5.0",
"tslib": "^2.3.0",
"zone.js": "~0.11.4"
},

```

Step #6: Then, import the **DiagramModule** into your **app.module.ts** file.

Refer to the code example.

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';

// import the DiagramModule for the Diagram component
import { DiagramModule } from '@syncfusion/ej2-angular-diagrams';

import { AppComponent } from './app.component';

@NgModule({
  //declaration of ej2-angular-diagrams module into NgModule
  imports: [ BrowserModule, DiagramModule ],
  declarations: [ AppComponent ],

```

 Copy

```
bootstrap: [ AppComponent ]
}))
export class AppModule { }
```

Step #7: Add the Syncfusion theme CDN reference links to the index.html file.

 Copy

```
<head>
.....
.....
.....

<link rel="icon" type="image/x-icon" href="favicon.ico">
<link href="//cdn.syncfusion.com/ej2/20.3.47/fluent.css" rel="stylesheet" />
<link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.
<link href="https://ej2.syncfusion.com/demos/src/diagram/styles/diagram-common.

.....
.....
.....

</head>
```

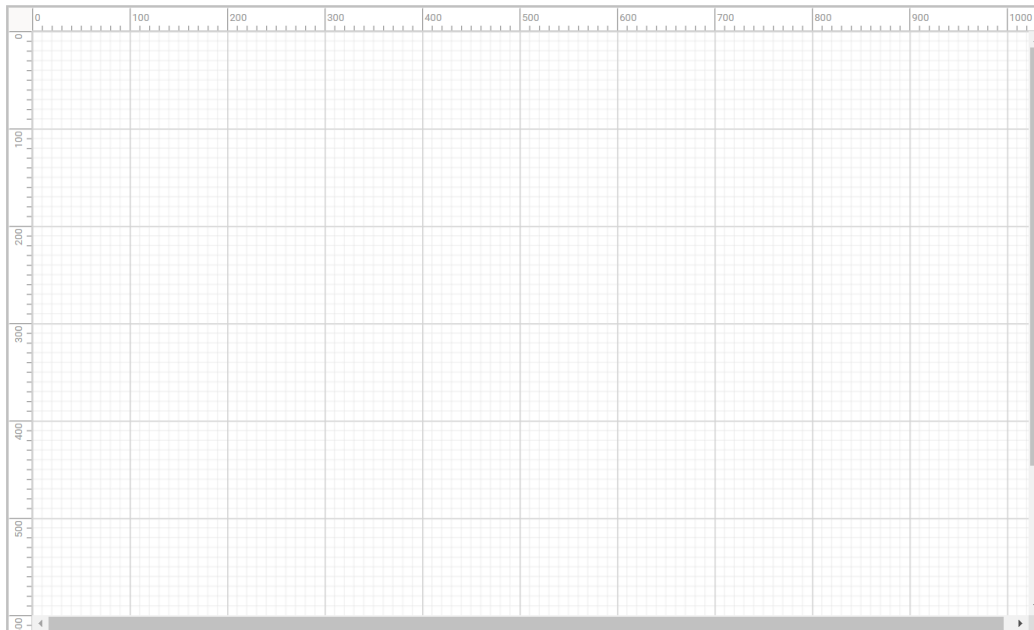
Step #8: Render the Diagram component utilizing the `<ejs-diagram>` selector in the template section of the `app.component.ts` file with the required width and height properties.



```
import { Component } from "@angular/core";

@Component({
  selector: "app-container",
  // specifies the template string for the diagram component
  template: `<ejs-diagram id="diagram" width="100%" height="100%px"></ejs-diagram>`
})
export class AppComponent {}
```

Step #9: Finally, run the app using the **npm start** command. Open the browser with the generated link to see an empty diagram as in the following image.



Creating the Diagram Canvas in the Angular Application

Create reusable BPMN symbols

The Angular Diagram control provides a gallery of reusable nodes and connectors called [SymbolPalettes](#). It displays a collection of palettes, each showing a set of nodes and connectors. We can drag and drop them onto the diagram canvas any number of times.

Follow these steps to create the Diagram symbol palette with BPMN shapes.

Step #1: First, initialize the Syncfusion `ejs- symbolpalette` tag by providing the necessary parameters, including width, height, and the assortment of symbols to be incorporated into the palette.

app.component.html

```
<div class="db-palette-parent">
  <!-- <div id="symbolpalette"></div> -->
  <ejs-symbolpalette id="symbolpalette"
    [expandMode]='expandMode' [palettes]='palettes'
    [enableSearch]="true">
```

[Copy](#)

```
width="100%" height="700px">  
</ejs-symbolpalette>  
</div>
```

Step #2: A palette is a feature that displays a group of related symbols, which can be textually annotated with a header. This collection of symbol groups can be added to the palette for easy access.

Use the **symbols** property to add predefined symbols to the palette, like in the following code example.

app.component.ts

```
public palettes: PaletteModel[] = [  
  {  
    id: 'bpmn_shapes',  
    expanded: true,  
    symbols: this.bpmnShapes,  
    iconCss: 'shapes',  
    title: 'BPMN Shapes'  
  },  
];
```

 Copy

Step #3: Now, define and customize the collection of symbols to be included in the palette according to your needs.



```
//Initialize bpmn shapes for symbol palette.
public bpmnShapes: NodeModel[] | any = [{
  id: 'Task',
  width: 35,
  height: 30,
  shape: {
    type: 'Bpmn',
    shape: 'Activity',
    activity: {
      activity: 'Task',
    },
  },
},
{
  id: 'Gateway',
  width: 30,
  height: 30,
  shape: {
    type: 'Bpmn',
    shape: 'Gateway',
  }
},
{
  id: 'Intermediate_Event',
  width: 30,
  height: 30,
```



```
shape: {
  type: 'Bpmn',
  shape: 'Event',
  event: {
    event: 'Intermediate'
  }
},
},
{
  id: 'End_Event',
  width: 30,
  height: 30,
  shape: {
    type: 'Bpmn',
    shape: 'Event',
    event: {
      event: 'End'
    }
  },
},
{
  id: 'Start_Event',
  width: 30,
  height: 30,
  shape: {
    type: 'Bpmn',
    shape: 'Event',
    event: {
      event: 'Start'
    }
  },
},
{

```

```
id: 'Collapsed_Sub-process',
width: 35,
height: 30,
shape: {
  type: 'Bpmn',
  shape: 'Activity',
  activity: {
    activity: 'SubProcess',
    subProcess: {
      collapsed: true,
      boundary: 'Default'
    }
  },
},
},
{
  id: 'Expanded_Sub-Process',
  width: 35,
  height: 30,
  constraints: NodeConstraints.Default | NodeConstraints.AllowDrop,
  shape: {
    shape: 'Activity',
    type: 'Bpmn',
    activity: {
      activity: 'SubProcess',
      subProcess: {
        type: 'Transaction',
        collapsed: false,
        processes: [],
        transaction: {
          cancel: {
            visible: false
          }
        }
      }
    }
  },
},
},
}
```

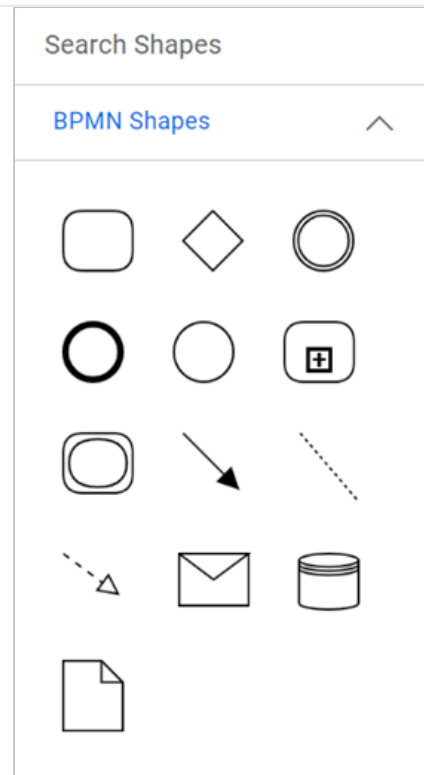
```
        failure: {
          visible: false
        },
        success: {
          visible: false
        }
      }
    }
  },
},
{
  id: 'Sequence_Flow',
  sourcePoint: {
    x: 0,
    y: 0
  },
  targetPoint: {
    x: 30,
    y: 30
  },
  type: 'Straight',
  targetDecorator: {
    shape: 'Arrow',
    style: {
      fill: 'black'
    }
  },
  shape: {
    type: 'Bpmn',
    flow: 'Sequence',
    sequence: 'Normal'
  },
},
```

```
},  
{  
  id: 'Association_Flow',  
  sourcePoint: {  
    x: 0,  
    y: 0  
  },  
  targetPoint: {  
    x: 30,  
    y: 35  
  },  
  type: 'Straight',  
  style: {  
    strokeDashArray: "2 2"  
  },  
  targetDecorator: {  
    shape: 'None'  
  },  
  sourceDecorator: {  
    shape: 'None'  
  },  
  shape: {  
    type: 'Bpmn',  
    flow: 'Association',  
    association: 'Default'  
  },  
},  
{  
  id: 'Message Flow',  
  sourcePoint: {  
    x: 0,  
    y: 0  
  },  
},
```

```
targetPoint: {
  x: 30,
  y: 22
},
type: 'Straight',
sourceDecorator: {
  shape: 'None'
},
targetDecorator: {
  shape: 'Arrow',
  style: {
    fill: 'white'
  }
},
style: {
  strokeDashArray: '4 4'
},
{
  id: 'Message',
  width: 35,
  height: 26,
  shape: {
    type: 'Bpmn',
    shape: 'Message',
  },
},
{
  id: 'Data_Source',
  width: 30,
  height: 28,
  shape: {
    type: 'Bpmn',
```

```
    shape: 'DataSource',  
  },  
},  
{  
  id: 'Data_Object',  
  width: 30,  
  height: 35,  
  shape: {  
    type: 'Bpmn',  
    shape: 'DataObject',  
    dataObject: {  
      collection: false,  
      type: 'None'  
    }  
  },  
},  
},  
];
```

Refer to the following image.



Creating a Gallery of Reusable BPMN Shapes Using Angular Diagram Control

Note: Refer to the [symbol palette in the Angular Diagram Library documentation](#) for more details.

Create the BPMN diagram

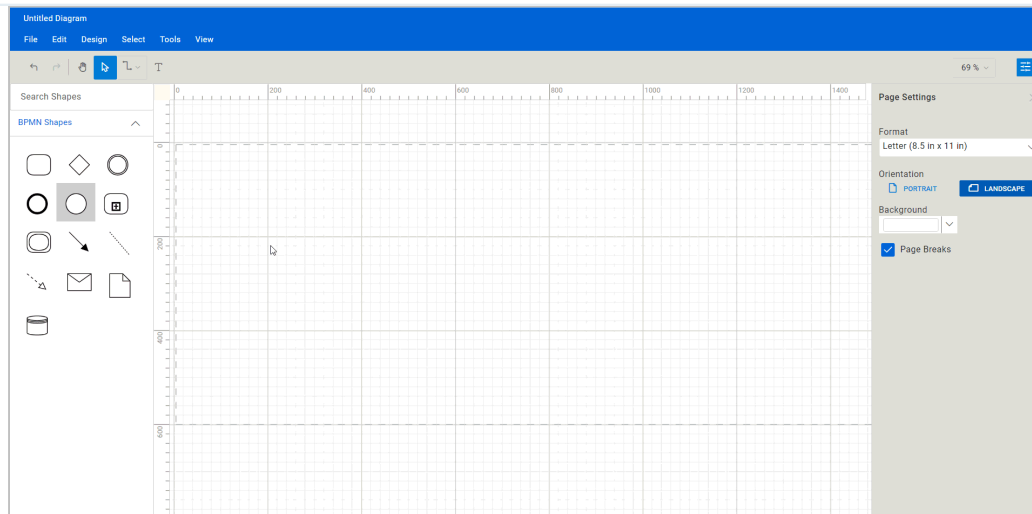
We can easily create the BPMN diagram now by dragging and dropping the BPMN symbols from the symbol palette and arranging and connecting them.

Add BPMN shapes to the editor

Drag symbols from the **BPMN Shapes** symbol palette onto the diagramming canvas to create a BPMN diagram.

The symbols represent various elements of a business process, such as tasks, events, and gateways. Arrange the symbols in a way that makes it clear where the start and end points of the business process are, as well as how the various nodes are connected. Organizing the symbols in a clear and logical order can help readers understand where to begin and how to follow the process flow.

Refer to the following GIF image.



Adding BPMN Symbols to the Canvas in the Angular Diagram Control

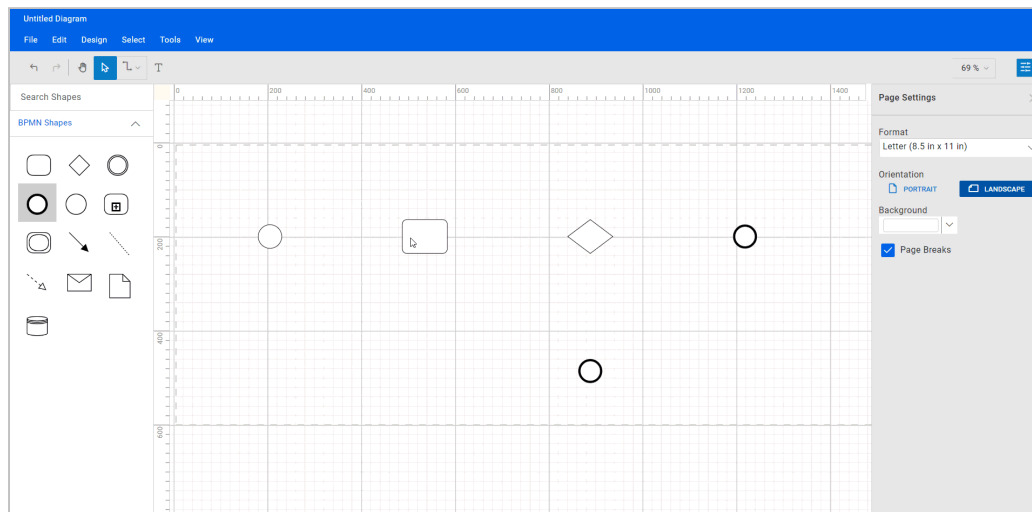
Selecting the appropriate symbols for each task or event is crucial in BPMN notation, as each symbol carries a specific meaning.

For instance, a task is denoted by a rectangle with rounded corners, while a circle symbolizes a start event. Utilizing the correct symbol ensures the diagram accurately mirrors the business process you aim to model.

In the Angular Diagram control, you can right-click on a BPMN shape, access the shortcut menu, and view or modify the shape's underlying data or attributes. This feature allows you to tailor a shape's appearance and its associated information.

For example, you can define the type of task to be performed using the **Task shape** property. The control offers several task types: **User**, **Service**, and **Receive**.

To alter the task type for a Task shape, right-click on the shape and select **Task Type** from the shortcut menu. Then, choose your preferred task type from the available options. The Task shape's appearance will then change to reflect your selection. For instance, a User task type will display a small sketch of a person in the upper left corner, indicating it is a user-defined task.



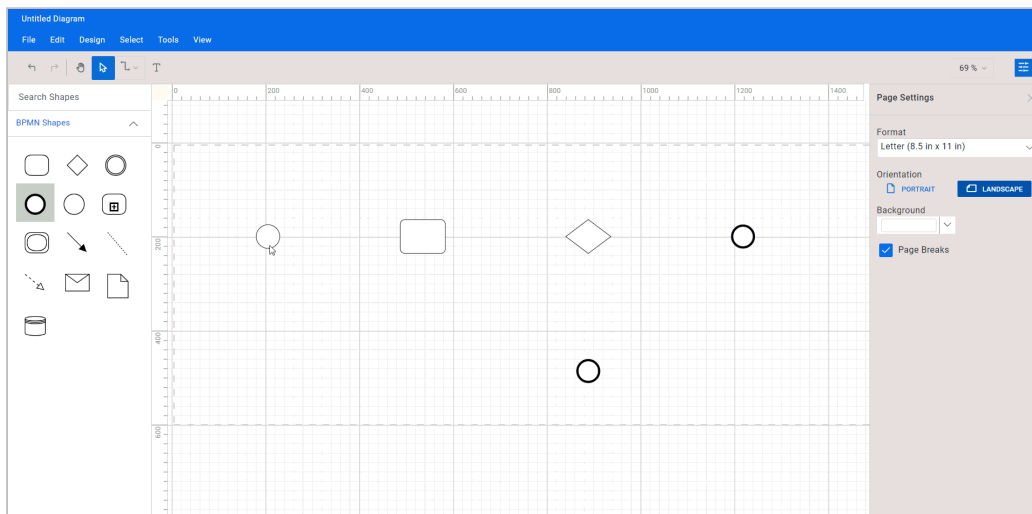
Change the BPMN Task Shape Type to User

Connect BPMN shapes

While creating a BPMN diagram, it is essential to connect diagram shapes to demonstrate the relationship between them. The Angular Diagram control has features that simplify this process using the **draw user handle** and **connector tools**.

The **draw user handle** is a small icon that will appear on each diagram shape when selected. You can use this feature to create a connection between two shapes quickly. To do so, click and drag the draw user handle from the first shape and drop it onto the second shape. A connector will automatically be created between the two shapes.

Refer to the following GIF image.

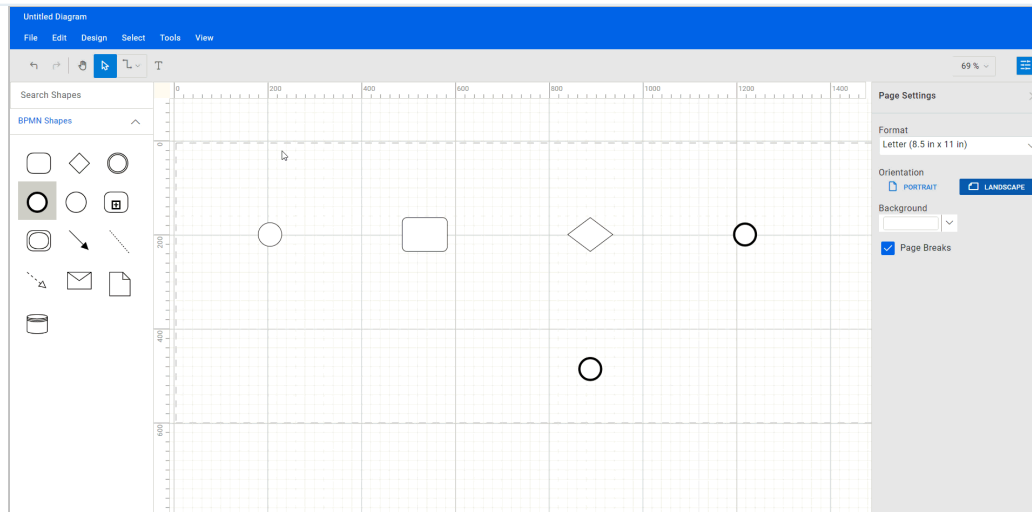


Connecting BPMN Shapes in BPMN Diagram Using the Draw User Handle Tool

Note: Refer to the [User handle in Angular Diagram control documentation](#) for more details.

If you prefer to use the **connector** tool, it is easily accessible from the toolbar or by selecting the **Tools -> Connector Tool** menu option. Select the tool, click on the first shape you want to connect, and then drag your mouse to the second shape. When you release the mouse, a connector will be created between the two shapes.

After creating the connectors, you can customize their appearance by changing the style, color, and thickness of the line. To select and move the shapes and connectors in your diagram, switch back to the **pointer tool** by clicking the pointer icon in the toolbar.

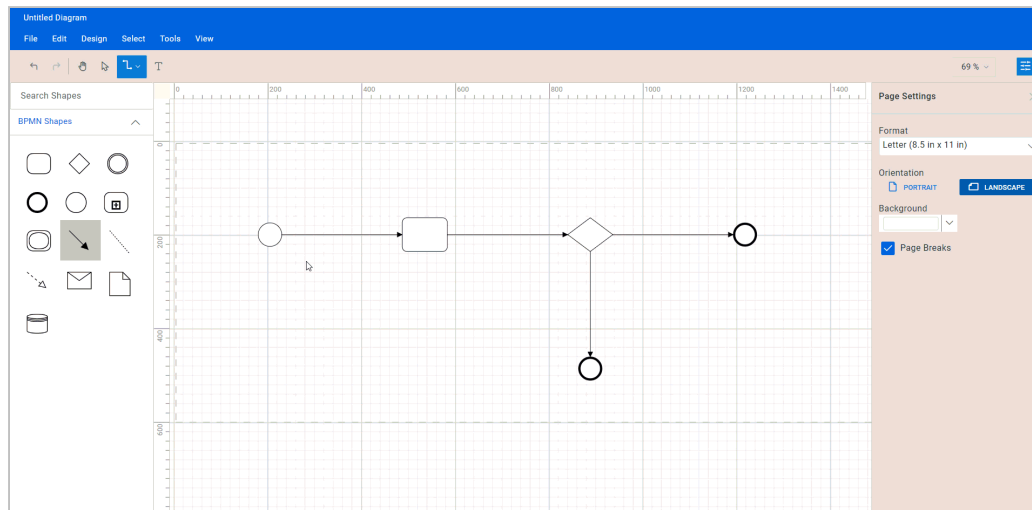


Creating Connectors Using Connector and Pointer Tools in Angular Diagram Control

Note: For more details, refer to the [Connector tools in the Angular Diagram control documentation](#).

All connectors created using the draw user handle or connector tool are sequence flow connectors by default. You can also change the connector type in the BPMN diagram by right-clicking the connector and selecting the desired type from the context menu.

For example, if you want to change a sequence flow connector to a message flow connector, right-click on the connector and select **Message Flow** from the options. Once you have changed the type of connector, the appearance of the connector will change to reflect the new type. A message flow connector will have an arrowhead at each end, whereas an association connector will have a dotted line.



Changing of Sequence Flow Connector to Message Flow Connector in BPMN Diagram

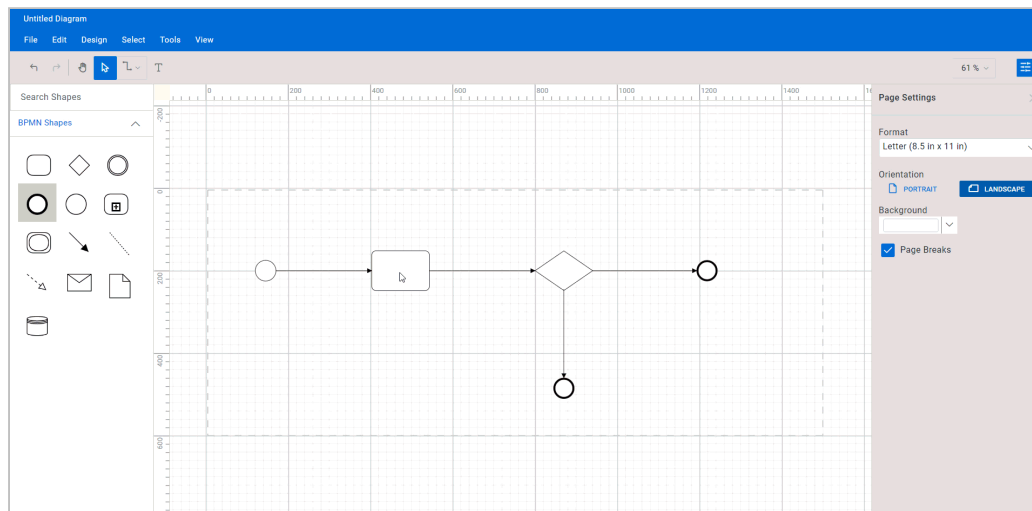
Note: Refer to the [BPMN flows in the Angular Diagram control documentation](#) for more details.

Add labels to BPMN shapes

After connecting the BPMN symbols, you can define the specific details of each process. One way to do this is by adding text to the symbols. This can provide additional information and make the diagram more informative.

To add text to a BPMN shape, **double-click** on the shape or select it and press **F2**. This will open a text box where you can type in any required text. After entering the text, press the **Esc** key or deselect the node to close the text box.

Refer to the following GIF image.



Adding Labels to BPMN Shapes in Angular Diagram Control

Note: Refer to the [Annotations in the Angular Diagram control documentation](#) for more details.

Load and save a diagram

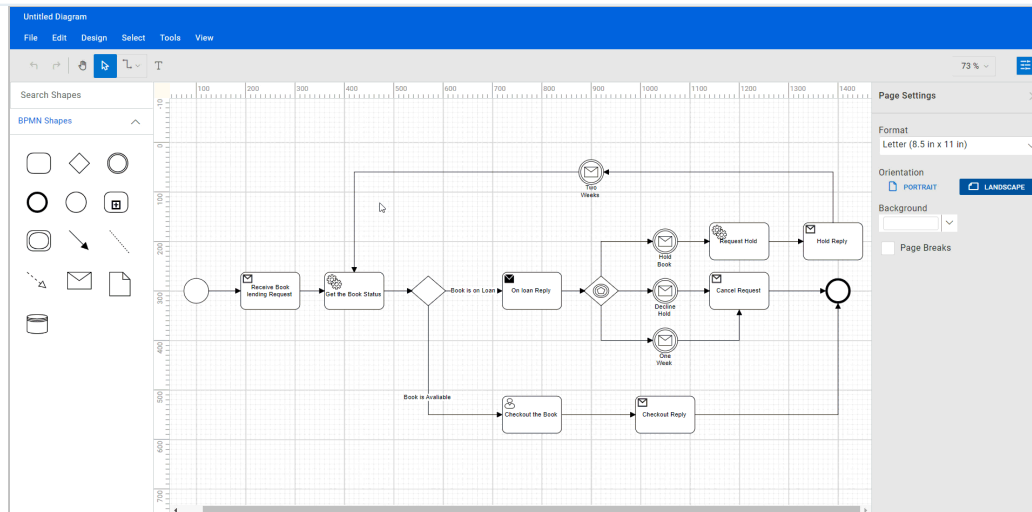
You can save your work and resume it later by loading the saved diagram back onto the diagram canvas.

To save your current diagram, go to the **File** menu and select **Save**, which will save your diagram as a file on your local drive.

To load an existing diagram file, go to the **File** menu and select **Open** to open a file dialog box. From there, you can search for and select the saved diagram file that you want to load.

This feature provides great flexibility and convenience, allowing you to pick up where you left off on a diagram or to make changes to a previously saved diagram.

Refer to the following GIF image.



Loading and Saving a BPMN Diagram in the Angular Diagram Control

Note: Refer to the [loading](#) and [saving diagrams in Angular Diagram](#) documentation for more details.

Undo or redo the diagram action

The Angular Diagram control supports undo and redo operations to track the changes made to a diagram. You can use these features to undo a mistake or restore an action you previously undid.

To undo an action, use the **Ctrl + Z** keyboard shortcut, click **Undo** on the toolbar, or use the **Undo** command from the **Edit** menu. This will undo the most recent action that was performed.

To redo an action that has been undone, you can use the **Ctrl + Y** keyboard shortcut, click **Redo** on the toolbar, or use the **Redo** command from the **Edit** menu. This will redo the most recently undone action.

Note: For more details, refer to the [Undo/redo in Angular Diagram control documentation](#).

Export the diagram

You can also export the diagram as an image file in **JPEG**, **PNG**, and **SVG** formats. By exporting the diagram as an image file, you can easily share it via email or other digital means or embed it in a document or presentation.

To do this, click the **File** menu and select **Export**. Once the export option is selected, a dialog box will appear where you can specify the name and file format of the image. You can also choose to export only the content area of the diagram

by excluding the blank areas or export the entire diagram (including blank areas) based on the width and height specified in the page settings.

Note: Refer to the [Exporting in Angular Diagram control documentation](#) for more details.

Print the diagram

To print a diagram, click the **File** menu and select **Print**. This will open the print dialog box where you can select your printer and customize the print settings, such as orientation, paper size, and page margins. Once you have configured the print settings to your preference, click **Print** to print the diagram.

Note: Refer to the [Printing in Angular Diagram control documentation](#) for more details.

Pan and zoom the diagram

The Angular Diagram control supports panning and zooming the diagram in different ways:

- ***Pan using the scrollbars:*** The most straightforward way to pan the diagram is by using the scrollbars on the right and bottom.
- ***Pan using the mouse wheel:*** You can also pan the diagram by using the mouse wheel. To scroll up or down, rotate the mouse wheel forward or backward; to scroll left or right, hold Shift while rotating the scroll wheel forward or backward.
- ***Pan using the Pan tool:*** You can pan the diagram by selecting the Pan tool from the toolbar. Then, hold down the left mouse button and drag the mouse to move the diagram in any direction.
- ***Zoom using keyboard shortcuts:*** The most efficient way to zoom in and out of the diagram is to use the **Ctrl + Mouse Wheel** shortcut.
- ***Zoom using the toolbar option:*** Another way to zoom in or out in a diagram is to use the zoom dropdown in the upper-right corner of the app window. In the Zoom dropdown, you can set the zoom level to a specific percentage or fit the entire diagram to the window.

References

Refer to [BPMN Viewer and Editor Using Angular Diagram Library demos on web](#) and [GitHub](#).

Conclusion

Thanks for reading! In this blog, we've explored how to create an interactive BPMN viewer and editor using the powerful and versatile features of the [Syncfusion Angular Diagram](#) control. With this tool, you can now easily demonstrate complex business workflows. Try it out and leave your feedback in the comments section below!

With Diagram, you can also create organizational charts, flowcharts, mind maps, floor plans, network diagrams, or logic circuit diagrams. Feel free to look at our [online examples](#) and [documentation](#) to explore the many other features and functionalities of the Syncfusion Angular Diagram control.

If you're an existing Syncfusion user, the newest version of Essential Studio® is available from the [License and Downloads](#) page. If you're new to Syncfusion, you can take advantage of our [30-day free trial](#) to explore the features and capabilities of our products.

For questions, you can contact us through our [support forum](#), [support portal](#), or [feedback portal](#). We're always happy to help you!

Related blogs

- [A Full-Stack Web App Using Angular and GraphQL: Part 1](#)
- [What's New in Angular Charts: 2023 Volume 3](#)
- [Build Multi-Parent Hierarchical Trees with Angular Diagram Library](#)
- [Creating Radial Tree Diagrams with Syncfusion Angular Diagram Library: A Complete Guide](#)



MEET THE AUTHOR

Moulidharan Gopalakrishnan

